Qt

- Sehr umfangreiches OpenSource GUI Toolkit
- Für unterschiedliche Platformen verfügbar
- Seit Version 4.5 lizenzkostenfrei unter LGPL verwendbar
- Leistungsfähige Entwicklungswerkzeuge verfügbar (QtCreator, Eclipse)
- Nun eine kleine Einführung...

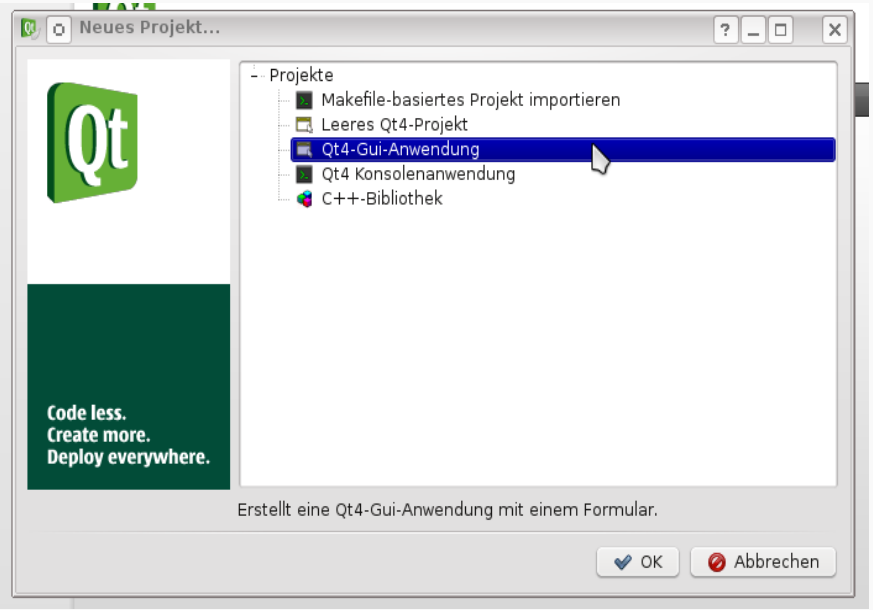
Installation und Start

- PC mit Linux, hier Ubuntu 9.10
- QtCreator >= 1.0.0, hier 1.2.1
- Die Paketverwaltung installiert Compiler usw.
- MuCross Toolchain für das Zielsystem

```
# . /usr/local/mucross/arm/environment-setup
# qtcreator
```

- „Neues Projekt“ klicken

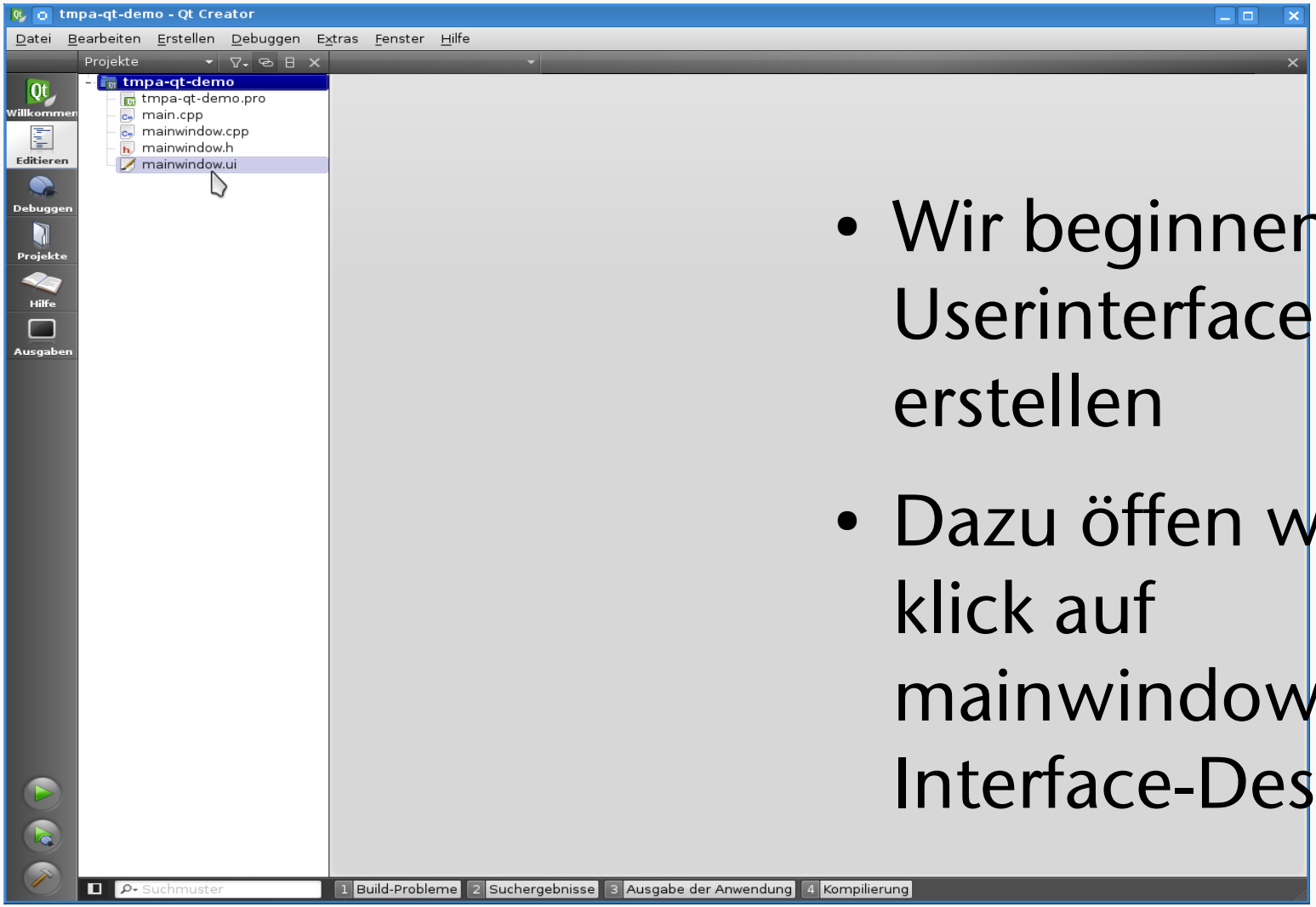
Projekt erstellen



- Wir wählen eine Qt4-Gui-Anwendung
- Geben ihr einen Namen
- Wählen noch das Qt-WebKit Modul aus
- Immer „weiter“, dann „abschließen“

kernel concepts

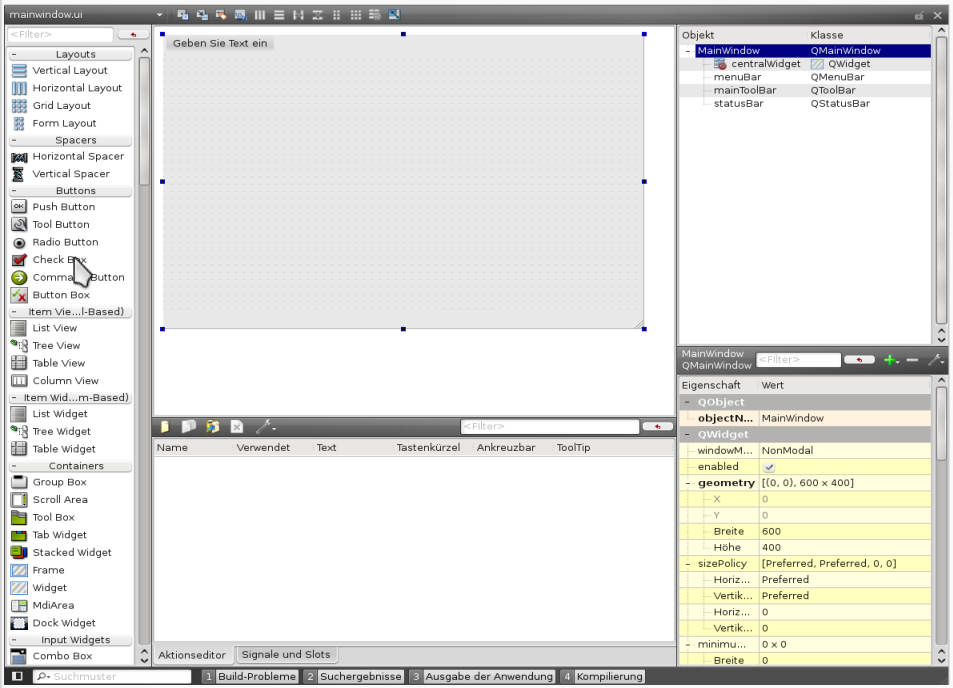
Leeres Projekt



- Wir beginnen, das Userinterface zu erstellen
- Dazu öffen wir mit klick auf mainwindow.ui den Interface-Designer

kernel concepts

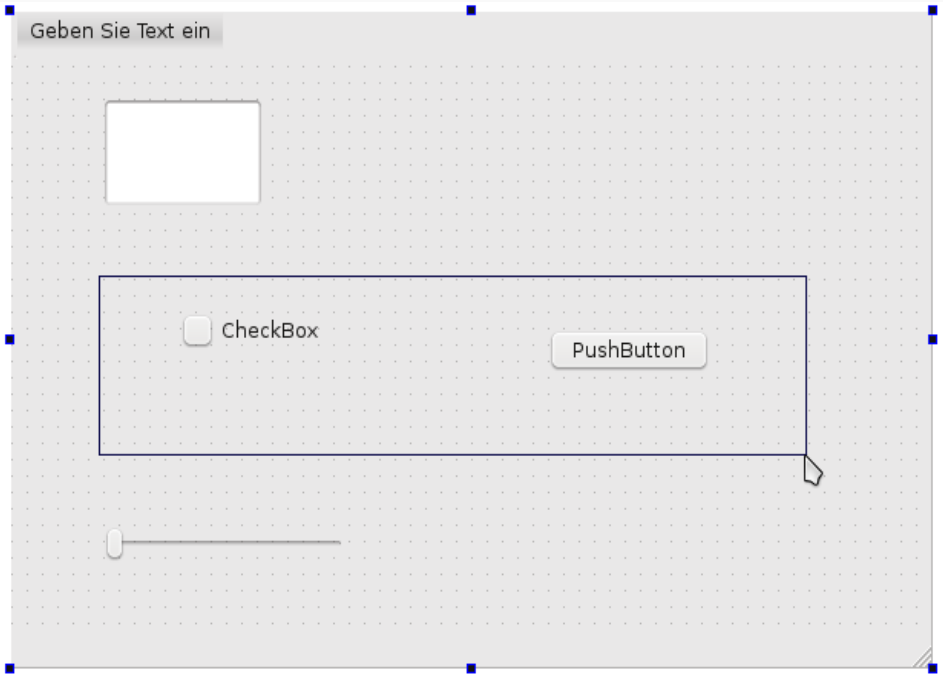
Leeres Interface

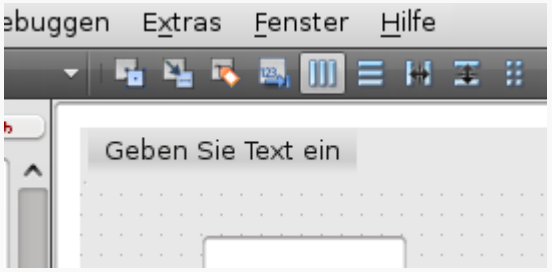


- Wir ziehen ein Text Edit, eine Check Box, einen Push Button und einen Horizontal Slider in das noch leere mittlere Feld

kernel concepts

Gruppieren (1)

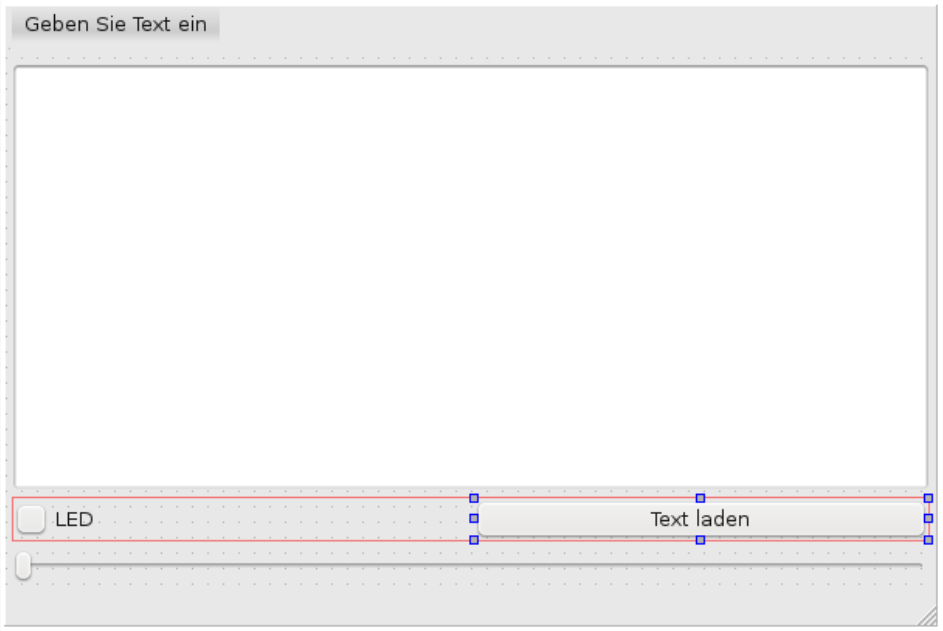




- Die beiden Knöpfe werden markiert, sie sollen nebeneinander liegen
- Klick auf das Symbol „waagerecht anordnen“

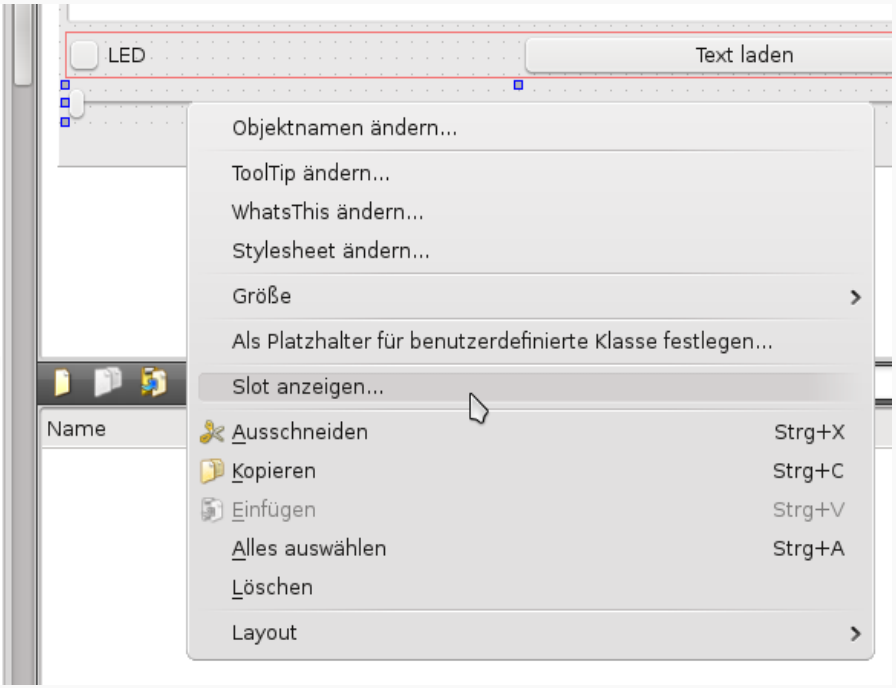
kernel concepts

Gruppieren (2)



- Jetzt rechts oben das Central Widget auswählen und das Symbol für „Objekte senkrecht anordnen“ klicken.
- Objekte anpassen
- Das Layout ist fertig

Beleben



- Wir verbinden einen Slot des Sliders, „ValueChanged(int)“
- QtCreator erzeugt automatisch den Code-Rumpf und wechselt in den Editor für die Datei mainwindow.cpp

kernel concepts

Fertiges Programm

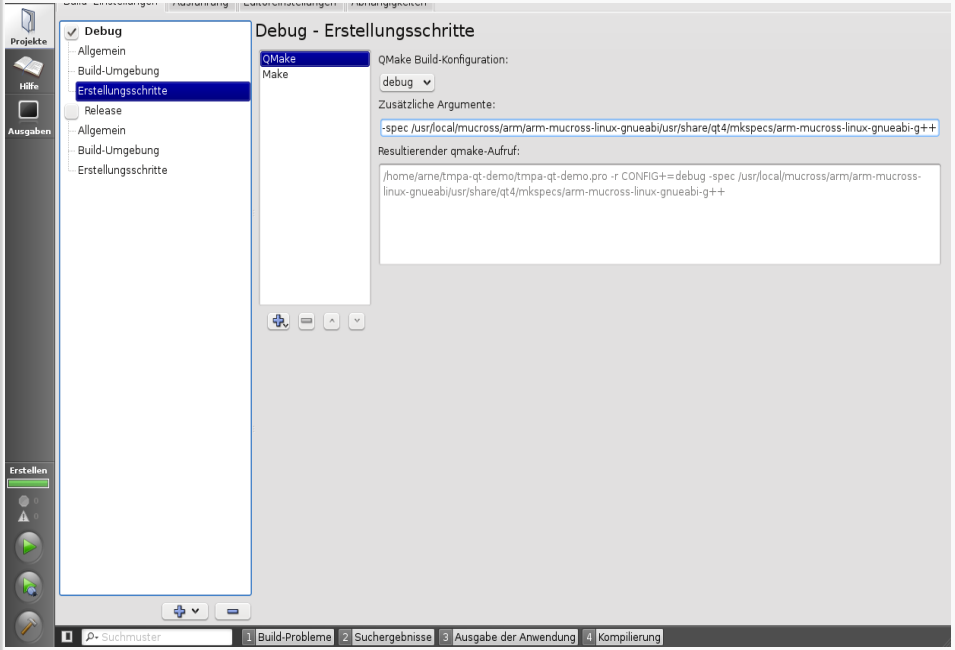
mainwindow.cpp <Symbol auswählen>

```
8
9   MainWindow::MainWindow(QWidget *parent)
10      : QMainWindow(parent), ui(new Ui::MainWindow)
11  {
12      ui->setupUi(this);
13  }
14
15  MainWindow::~MainWindow()
16  {
17      delete ui;
18  }
19
20  void MainWindow::on_horizontalSlider_valueChanged(int value)
21  {
22      stringstream s;
23      s << "echo " << value << "> /sys/devices/platform/led-topas/led_segment";
24      system(s.str().c_str());
25  }
26
27  void MainWindow::on_checkBox_clicked(bool checked)
28  {
29      if(checked)
30          system("echo 1 > /sys/devices/platform/led-topas/led");
31      else
32          system("echo 0 > /sys/devices/platform/led-topas/led");
33  }
34
35  void MainWindow::on_pushButton_clicked()
36  {
37      QFile file("utf8.txt");
38      if (!file.open(QIODevice::ReadOnly | QIODevice::Text))
39           return;
40
41      while (!file.atEnd()) {
42          QByteArray line = file.readLine();
43          ui->textEdit->append(QString::fromUtf8(line));
44      }
45
46  }
47
```

- Noch läuft alles auf dem Entwicklungs-PC
- Für einen Testlauf einfach die grüne „Play“-Taste drücken
- QtCreator unterstützt bei der Fehlersuche

kernel concepts

Cross Compilieren



- Qmake -spec eingeben bei „Projekte-Erstellungsschritte-Qmake-Zusätzliche Argumente“:

-spec /usr/local/mucross/arm/arm-mucross-linux-gnueabi/usr/share/qt4/mkspecs/arm-mucross-linux-gnueabi-g++

kernel concepts

Übertragung auf das Target

- „Erstellen-Alles neu Erstellen“ baut das Projekt jetzt für die Zielhardware
- Das Programm tmpa-qt-demo/tmpa-qt-demo ist auf dem Target direkt lauffähig
- QtCreator unterstützt noch kein Remote-Debugging
- Mit ein paar Tricks geht es trotzdem

QtCreator

- QtCreator bietet viele Tastenkürzel, die die Arbeit erleichtern
- Der „Locator“ erlaubt schnelles Navigieren auch durch große Projekte
- Gute Einführungsvideos gibt es bei youtube!

Ende

- Der komplette Quelltext ist mit auf der DVD enthalten

Vielen Dank!